

ادیب رضائی - امیرمحمد خسروی	نام و نام خانوادگی
810198386 - 810198401	شماره دانشجویی
1401.04.10	تاریخ ارسال گزارش



به نام خدا
 دانشگاه تهران
 دانشکده مهندسی
 برق و کامپیوتر



درس شبکه‌های عصبی و یادگیری عمیق

تمرین ششم

فهرست

1

پاسخ ۱. تشخیص شبکه های رمزگذار رمزگشای مولد

1

۱-۱- توضیحات مدل ها

2

مدل پیشنهادی مقاله:

3

پیش پردازش داده

4

نتایج

8

پاسخ ۲ - تشخیص خشونت در فیلم

شكل‌ها

.1.1

جدول‌ها

پاسخ 1. شبکه های رمزگذار-رمزگشا مولد

1-1- مجموعه دادگان مقاله

ابتدا داده ها را میخوانیم و به دو بخش train و test تقسیم میکنیم. ما از داده های fashion_mnist استفاده میکنیم. سپس چند تصویر رندوم از مجموعه داده ها به همراه برچسب آن ها را نمایش میدهیم. نتیجه به صورت زیر است:



شکل 1-1. چند نمونه رندوم به همراه برچسب

تعداد و شکل تصاویر به صورت زیر است:

```
data.explore()
```

```
Number of train samples = 60000
Train samples shape = (60000, 28, 28)
Number of test samples = 10000
Test samples shape = (10000, 28, 28)
```

شکل 1-2. تعداد و شکل تصاویر قبل پیش پردازش

حال روی داده های train و test پیش پردازش انجام میدهیم. مراحل preprocessing به این صورت است که ابتدا داده ها را standardize میکنیم. به این معنا که میانگین مقادیر را 0 و واریانس آن ها را 1

میکنیم. سپس داده ها را normalize میکنیم. به این معنا که همه داده ها در محدوده 0 تا 1 قرار گیرند. در نهایت نیز عکس ها را flatten میکنیم تا یک بعد داشته باشند. همین کار را با label ها میکنیم.

تعداد و شکل داده ها پس از پیش پردازش به صورت زیر خواهد بود:

```
data.explore()
```

```
Number of train samples = 60000
Train samples shape = (60000, 784)
Number of test samples = 10000
Test samples shape = (10000, 784)
```

شکل 1-3. تعداد و شکل تصاویر بعد از پیش پردازش

1-2- انجام PCA و ISOMAP

PCA و ISOMAP دو الگوریتم معروف در حوزه کاهش ابعاد داده ها هستند. هر کدام از این الگوریتم ها رویکرد متفاوتی برای تبدیل داده ها به فضای کمتر ابعاد دارند. در ادامه به شرح هر الگوریتم پرداخته و خلاصه ای از مزایا و معایب آنها را ارائه می دهم:

:PCA-Principal Component Analysis .1

- عملکرد: PCA به منظور تبدیل مجموعه داده ها از فضای بزرگتر ابعاد به فضای کوچکتر ابعاد استفاده می شود. این الگوریتم با استفاده از تحلیل مقادیر ویژه ماتریس کوواریانس داده ها، بردارهای ویژه (principal components) را استخراج می کند. این بردارهای ویژه به ترتیب اهمیت در توضیح داده ها (واریانس آنها) مرتب می شوند.

- مزایا:

- کاهش ابعاد: PCA قادر است به طور موثر ابعاد داده ها را کاهش دهد و اطلاعات اصلی داده ها را حفظ کند.

- رابطه با ویژگی‌ها: PCA به دنبال بردارهای ویژه است که با ویژگی‌های اصلی داده‌ها بیشترین همبستگی را دارند.

- معایب:

- خطر اطلاعات از دست رفته: با کاهش ابعاد، برخی از اطلاعات دقیق و جزئی‌تر داده‌ها ممکن است از دست بروند.

- آسیب پذیری نسبت به داده‌های پرت: PCA حساس به داده‌های پرت است و می‌تواند در تحلیل و تبدیل داده‌ها به مشکل برخورد کند.

:ISOMAP .2

- عملکرد: ISOMAP یک الگوریتم کاهش ابعاد غیرخطی است که بر مبنای یافتن نزدیک‌ترین همسایه

ای هر داده و ساختن یک نمودار گراف از روی آنها عمل می‌کند. این الگوریتم از یک روش مبتنی بر گراف به نام "کوتاه‌ترین مسیرهای جهانی" (shortest path distances) استفاده می‌کند تا فضای میانگینی (manifold) داده‌ها را در فضای کمتر ابعاد بازسازی کند.

- مزایا:

- تشخیص ساختار غیرخطی: ISOMAP قادر است به طور خوب ساختار غیرخطی داده‌ها را تشخیص دهد و نمایشی کم‌ابعاد از آنها بسازد.

- مقاومت به داده‌های پرت: ISOMAP نسبتاً مقاوم در برابر داده‌های پرت است و می‌تواند در تبدیل داده‌های پرت به فضای کمتر ابعاد بهتر عمل کند.

- معایب:

- پیچیدگی محاسباتی: ساختن گراف در ISOMAP نیاز به محاسبات پیچیده و منابع محاسباتی بیشتری دارد.

- وابستگی به نزدیک‌ترین همسایه: انتخاب درست نزدیک‌ترین همسایه‌ها می‌تواند تأثیر زیادی در عملکرد ISOMAP داشته باشد.

به طور کلی، PCA بیشتر برای کاهش ابعاد خطی و تشخیص ویژگی‌ها مناسب است. اما ISOMAP برای کاهش ابعاد غیرخطی و بازسازی ساختار غیرخطی داده‌ها مناسب‌تر است. هر کدام از این الگوریتم‌ها مزايا و معایب خود را دارند و انتخاب بین آنها بستگی به ماهیت داده‌ها و هدف مورد نظر دارد.

از دو متده که شده استفاده میکنیم. نتیجه به ازای تعداد کامپوننت های 35 و 27 به صورت زیر خواهد بود:

```
Method PCA : Accuracy = 0.8469 Number of components = 40  
Method PCA : Accuracy = 0.8441 Number of components = 30  
Method ISOMAP : Accuracy = 0.7963 Number of components = 40  
Method ISOMAP : Accuracy = 0.7947 Number of components = 30
```

شكل 1-4. نتایج دو مدل

سپس از استفاده میکنیم تا بهترین پارامترها را به دست آوریم. نتیجه به صورت زیر است:

```
PCA Best Parameter: 35  
ISOMAP Best Parameter: 27  
KNN Best Parameter: 19
```

شكل 1-5. بهترین پارامترها

با در نظر گرفتن این پارامترها نتیجه به صورت زیر میشود:

```
Method PCA : Accuracy = 0.8421 Number of components = 35  
Method PCA : Accuracy = 0.8368 Number of components = 27  
Method ISOMAP : Accuracy = 0.7939 Number of components = 35  
Method ISOMAP : Accuracy = 0.7937 Number of components = 27
```

شكل 1-6. نتایج دو مدل (2)

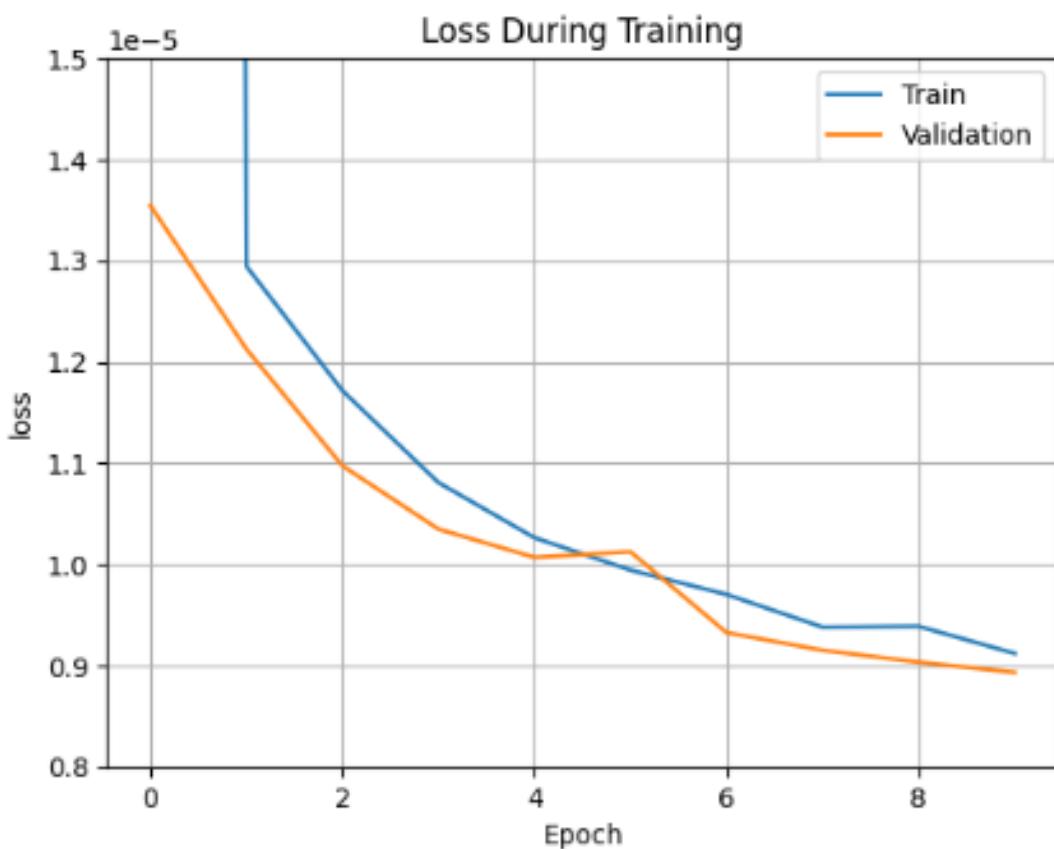
۱-۳- رمزگذار-رمزگشا

ابدا با استفاده از Dense Layer ها و با Latent-Dimension 100 رمزگذار-رمزگشا را میسازیم و آن را روی داده ها آموزش میدهیم و سپس از قسمت رمزگذار برای کاهش بعد استفاده میکنیم و فضای کاهش بعد یافته را با KNN طبقه بندی میکنیم. نتیجه به صورت زیر است:

معماری شبکه:

Layer (type)	Output Shape	Param #
<hr/>		
input_12 (InputLayer)	[(None, 28, 28)]	0
flatten_11 (Flatten)	(None, 784)	0
dense_74 (Dense)	(None, 784)	615440
batch_normalization_56 (BatchNormalization)	(None, 784)	3136
re_lu_56 (ReLU)	(None, 784)	0
dense_75 (Dense)	(None, 392)	307720
batch_normalization_57 (BatchNormalization)	(None, 392)	1568
re_lu_57 (ReLU)	(None, 392)	0
dense_76 (Dense)	(None, 196)	77028
batch_normalization_58 (BatchNormalization)	(None, 196)	784
re_lu_58 (ReLU)	(None, 196)	0
dense_77 (Dense)	(None, 100)	19700
dense_78 (Dense)	(None, 196)	19796
batch_normalization_59 (BatchNormalization)	(None, 196)	784
re_lu_59 (ReLU)	(None, 196)	0
dense_79 (Dense)	(None, 392)	77224
batch_normalization_60 (BatchNormalization)	(None, 392)	1568
re_lu_60 (ReLU)	(None, 392)	0
dense_80 (Dense)	(None, 784)	308112
<hr/>		
Total params: 1,432,860		
Trainable params: 1,428,940		
Non-trainable params: 3,920		

:val_loss و loss نمودار



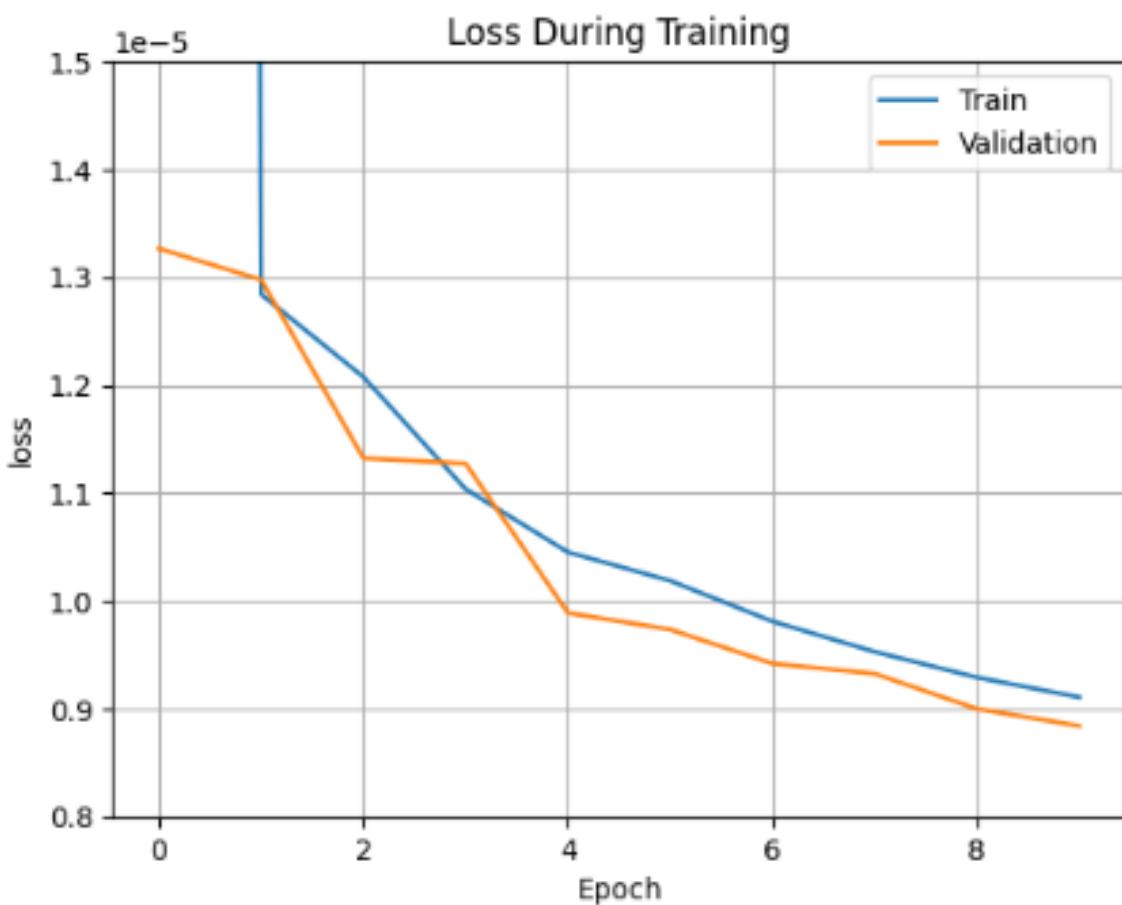
:دقت مدل:

```
1688/1688 [=====] - 6s 4ms/step
313/313 [=====] - 1s 4ms/step
Autoencoder with latent dim = 100 plus KNN Accuracy = 0.8139
```

همانطور که مشاهده میشود دقت در این حالت 8139 درصد است.

سپس همان معماری را اما این بار با latent_dimension برابر 50 تکرار میکنیم و مجددا آن را روی داده ها آموزش داده و با کمک KNN یک طبقه بند میسازیم. این بار نتایج به صورت زیر است:

:val_loss و loss نمودار



دقت مدل:

```
1688/1688 [=====] - 6s 4ms/step
313/313 [=====] - 1s 4ms/step
Autoencoder with latent dim = 100 plus KNN Accuracy = 0.8126
```

همانطور که مشاهده میشود دقت مدل در این حالت 81.26 درصد است که تفاوت چندانی با حالت قبل ندارد (مقدار ناچیزی کمتر است).

حال با استفاده از Convolution Layer ها رمزگذار-رمزگشا میسازیم و آن را روی مجموعه دادگان آموزش میدهیم و سپس از قسمت رمزگذار برای کاهش بعد استفاده میکنیم و فضای کاهش بعد یافته را با KNN طبقه بندی کرده و همچنین دو مقدار متفاوت برای latent_dimension تست میکنیم (مقدار اول بزرگتر و مقدار دوم کوچک تر است). نتایج به صورت زیر خواهد بود:

معماری مدل:

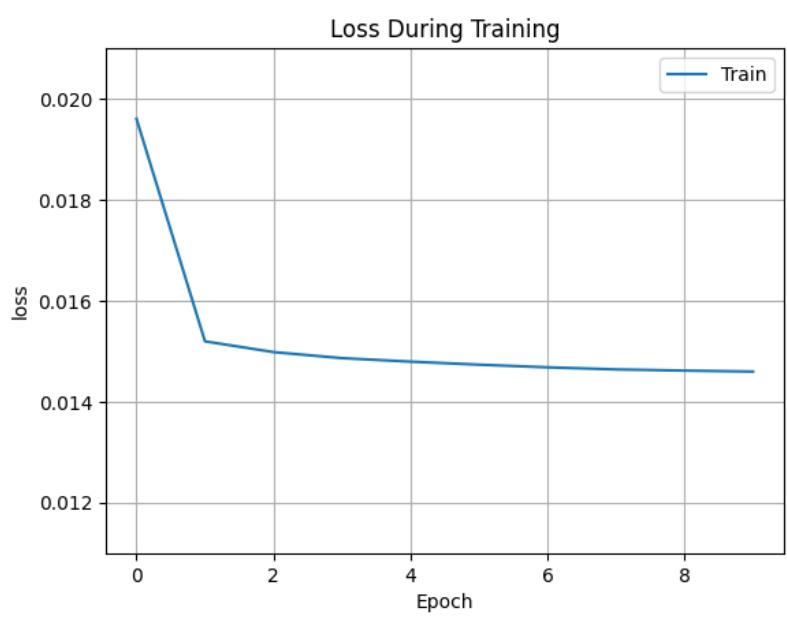
```

Model: "model_43"

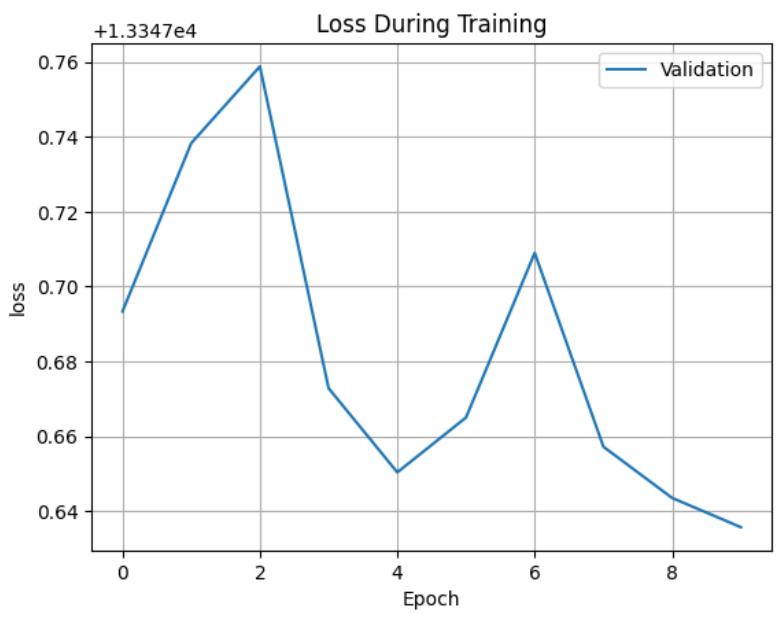
Layer (type)          Output Shape         Param #
=====
input_32 (InputLayer)     [(None, 28, 28, 1)]      0
conv2d_47 (Conv2D)        (None, 14, 14, 16)    160
conv2d_48 (Conv2D)        (None, 7, 7, 8)       1160
conv2d_transpose_31 (Conv2D Transpose) (None, 14, 14, 8) 584
conv2d_transpose_32 (Conv2D Transpose) (None, 28, 28, 16) 1168
conv2d_49 (Conv2D)        (None, 28, 28, 1)      145
=====
Total params: 3,217
Trainable params: 3,217
Non-trainable params: 0

```

:train روی داده های نمودار loss



نمودار loss روی داده های validation:



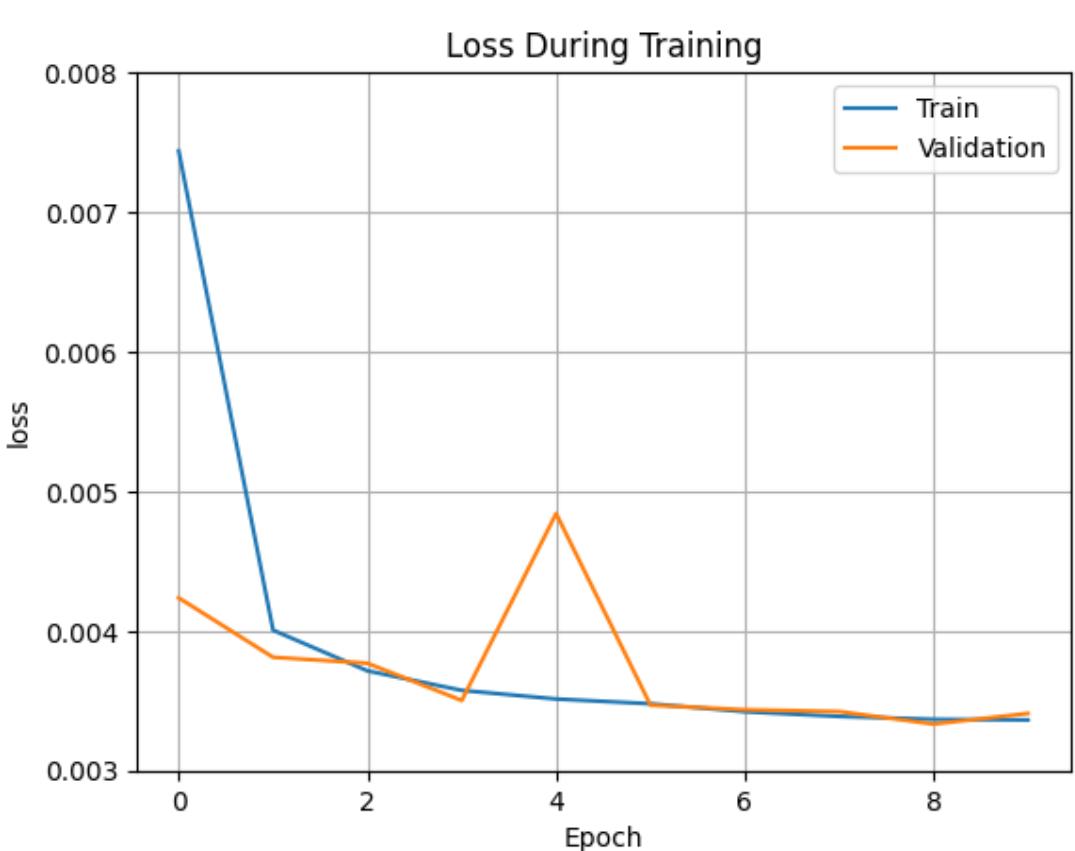
:دقت مدل

```
(54000, 784)
1688/1688 [=====] - 3s 2ms/step
(54000, 7, 7, 8)
313/313 [=====] - 1s 2ms/step
(10000, 7, 7, 8)
Autoencoder with latent dim = 100 plus KNN Accuracy = 0.8393
```

دقت همانطور که مشاهده میشود، مقدار 83.93 درصد است.

حال برای حالتی که latent_dimension کوچک تر است داریم:

نمودارهای loss و val_loss



دقت مدل:

```
(54000, 784)
1688/1688 [=====] - 3s 2ms/step
(54000, 7, 7, 8)
313/313 [=====] - 1s 2ms/step
(10000, 7, 7, 8)
Autoencoder with latent dim = 100 plus KNN Accuracy = 0.8442
```

همانطور که مشاهده میشود دقت 84.42 درصد است که دقتهای از مرحله قبل میباشد. به این معنا که لزوماً کاهش اندازه ابعاد فضای latent باعث کاهش دقتنمیشود و گاهی نیز باعث افزایش آن خواهد شد.

۱-۴- خود رمزگذار متغیر (Variational AutoEncoder)

یک مدل مولد احتمالاتی است که برای کاهش ابعاد و تولید داده‌های جدید استفاده می‌شود. در ادامه خلاصه‌ای از شیوه عملکرد VAE و مزایا آن را ارائه می‌دهم:

- شیوه عملکرد: VAE از دو بخش اصلی تشکیل شده است: encoder و decoder. Encoder شبکه‌ای عمیق است که داده‌ها را به فضای نهان (latent space) با ابعاد کمتر نگاشت می‌کند. این فضای نهان معمولاً یک توزیع احتمالی گاووسی است. سپس، با استفاده از توزیع احتمالی فضای نهان، نمونه‌های تصادفی تولید می‌شود و توسط decoder به فضای داده اصلی بازسازی می‌شوند. VAE با ترکیب این دو بخش، مدل آموزش می‌دهد که بتواند فضای نهان را یاد بگیرد و داده‌ها را در این فضا نمایش دهد.

- مزایا:

- تولید داده جدید: VAE قادر به تولید داده‌های جدید است که از توزیع احتمالی فضای نهان ساخته شده‌اند. این قابلیت به عنوان یک مولد داده مفید است.
- کاهش ابعاد: VAE به صورت همزمان یاد می‌گیرد که ابعاد داده‌ها را کاهش دهد و یک فضای نهان ساختارمند را تعلیم بدهد.
- ترکیبی از خلاقیت و انعطاف‌پذیری: VAE با توزیع احتمالی فضای نهان، خلاقیت بالا در تولید داده‌های جدید دارد و می‌تواند به تفسیرات متفاوتی از داده‌ها منجر شود.

- معایب:

- دشواری در آموزش: VAE آموزش مدلی پیچیده است و نیاز به محاسبه گرادیان‌ها در فضای نهان دارد که ممکن است زمانبر باشد.

ابتدا با استفاده از Convolution Layer ها و سپس با استفاده از Dense Layer ها Variational AutoEncoder می‌سازیم و آن را روی مجموعه دادگان آموزش میدهیم و سپس از قسمت رمزگذار برای کاهش بعد استفاده می‌کنیم و فضای کاهش بعد یافته را با KNN طبقه بندی می‌کنیم و همچنین پدو مقدار متفاوت 10 و 2 را برای latent_dimension تست می‌کنیم و نتایج را گزارش کنید.

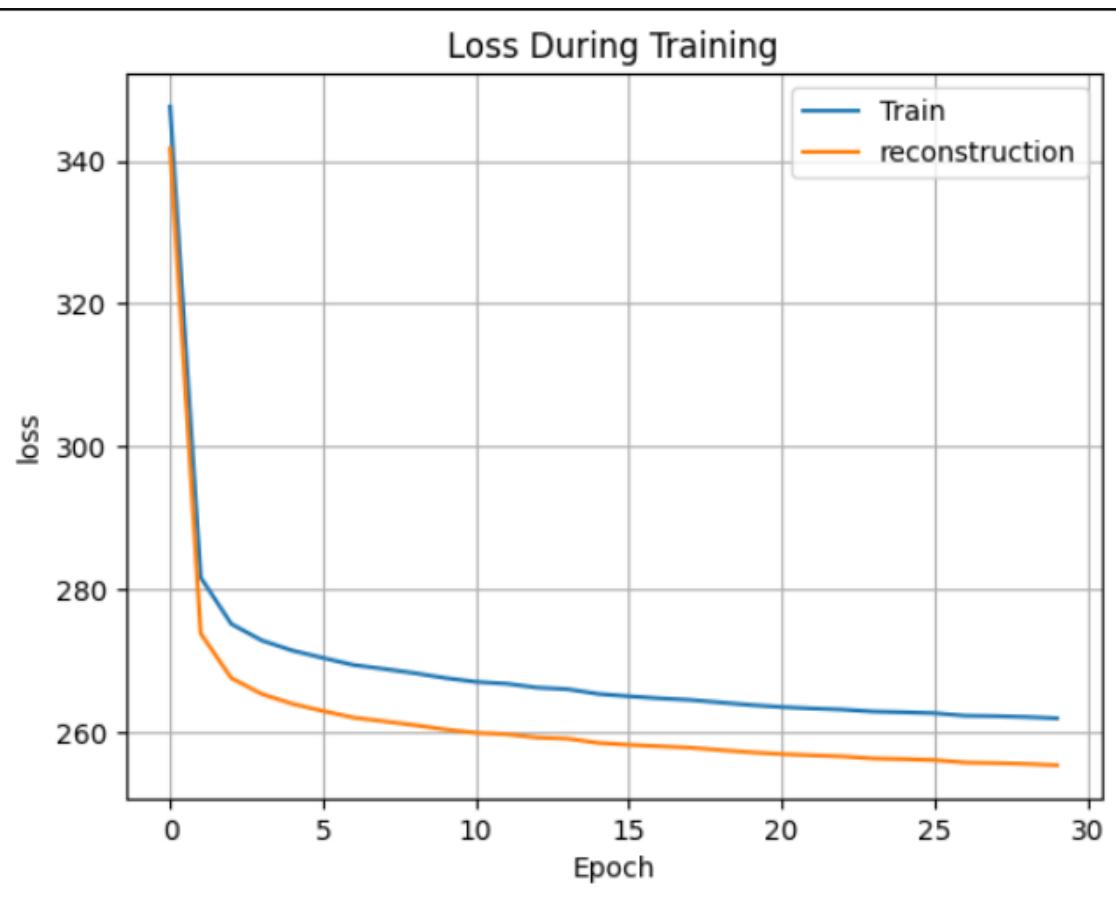
- معماری مدل:

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
input_42 (InputLayer)	[(None, 28, 28, 1)]	0	[]
conv2d_64 (Conv2D)	(None, 14, 14, 32)	320	['input_42[0][0]']
conv2d_65 (Conv2D)	(None, 7, 7, 64)	18496	['conv2d_64[0][0]']
flatten_17 (Flatten)	(None, 3136)	0	['conv2d_65[0][0]']
dense_95 (Dense)	(None, 16)	50192	['flatten_17[0][0]']
z_mean (Dense)	(None, 2)	34	['dense_95[0][0]']
z_log_var (Dense)	(None, 2)	34	['dense_95[0][0]']
sampling_4 (Sampling)	(None, 2)	0	['z_mean[0][0]', 'z_log_var[0][0]']
<hr/>			
Total params: 69,076			
Trainable params: 69,076			
Non-trainable params: 0			
<hr/>			
Model: "decoder"			
<hr/>			
Layer (type)	Output Shape	Param #	
input_43 (InputLayer)	[(None, 2)]	0	
dense_96 (Dense)	(None, 3136)	9408	
reshape_3 (Reshape)	(None, 7, 7, 64)	0	
conv2d_transpose_46 (Conv2D (None, 14, 14, 64) Transpose)	(None, 28, 28, 32)	36928	
conv2d_transpose_47 (Conv2D (None, 28, 28, 32) Transpose)	(None, 28, 28, 1)	18464	
conv2d_transpose_48 (Conv2D (None, 28, 28, 1) Transpose)		289	
<hr/>			
Total params: 65,089			
Trainable params: 65,089			
Non-trainable params: 0			

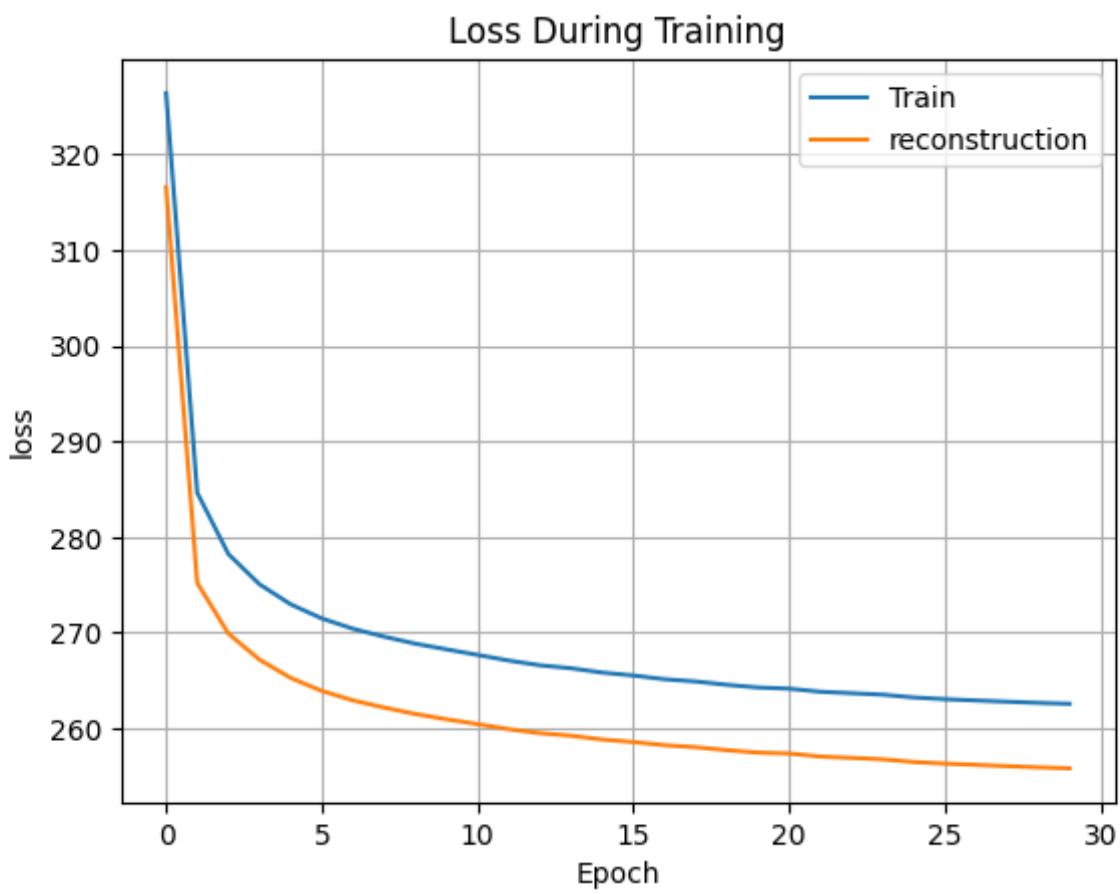
در یک حالت از convolutional layer ها و در حالت دیگر از dense layer ها استفاده میکنیم.

ابتدا با latent_dimension برابر 2 مراحل را انجام میدهیم. نتایج به صورت زیر است:

:val_loss و loss نمودار



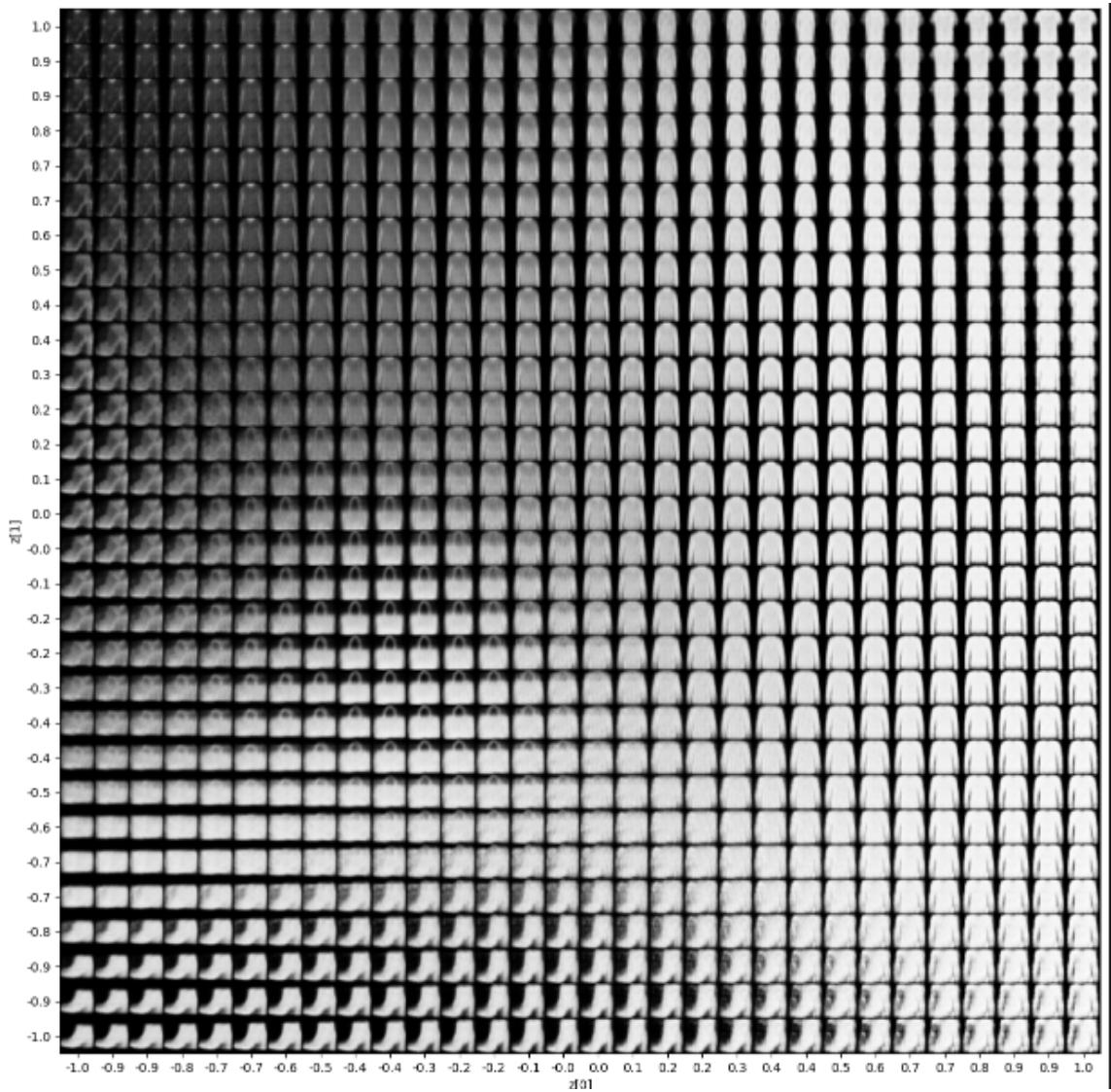
سپس با latent_dimension برابر با 10 مراحل را انجام میدهیم و نتایج به صورت زیر است:
نمودار val_loss و loss



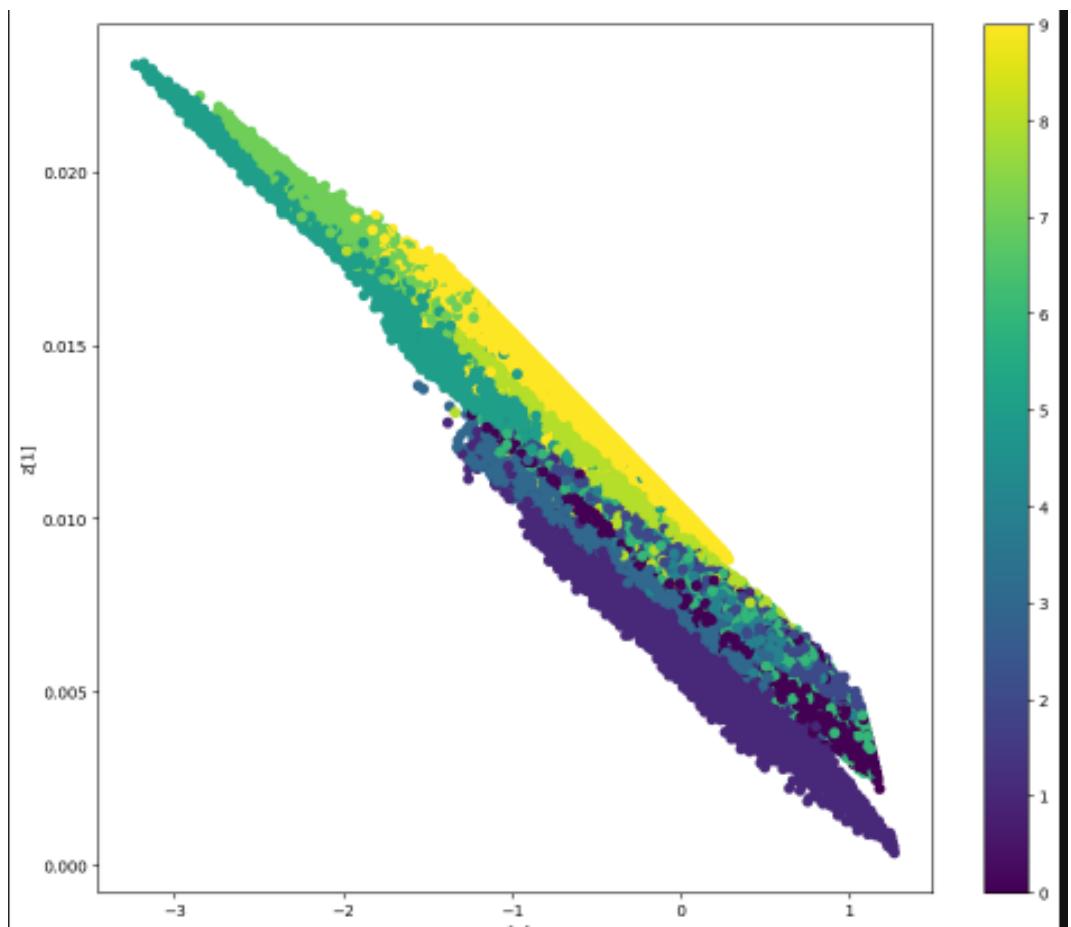
۱-۵- کاوش در فضای latent

با استفاده از grid np.linspace میسازیم و به شبکه VAE Decoder میدهیم و تصاویر زیر تولید میشوند. همچنین داده Train را به شبکه Encoder میدهیم و plot داده آموزشی را در فضای latent نمایش میدهیم.

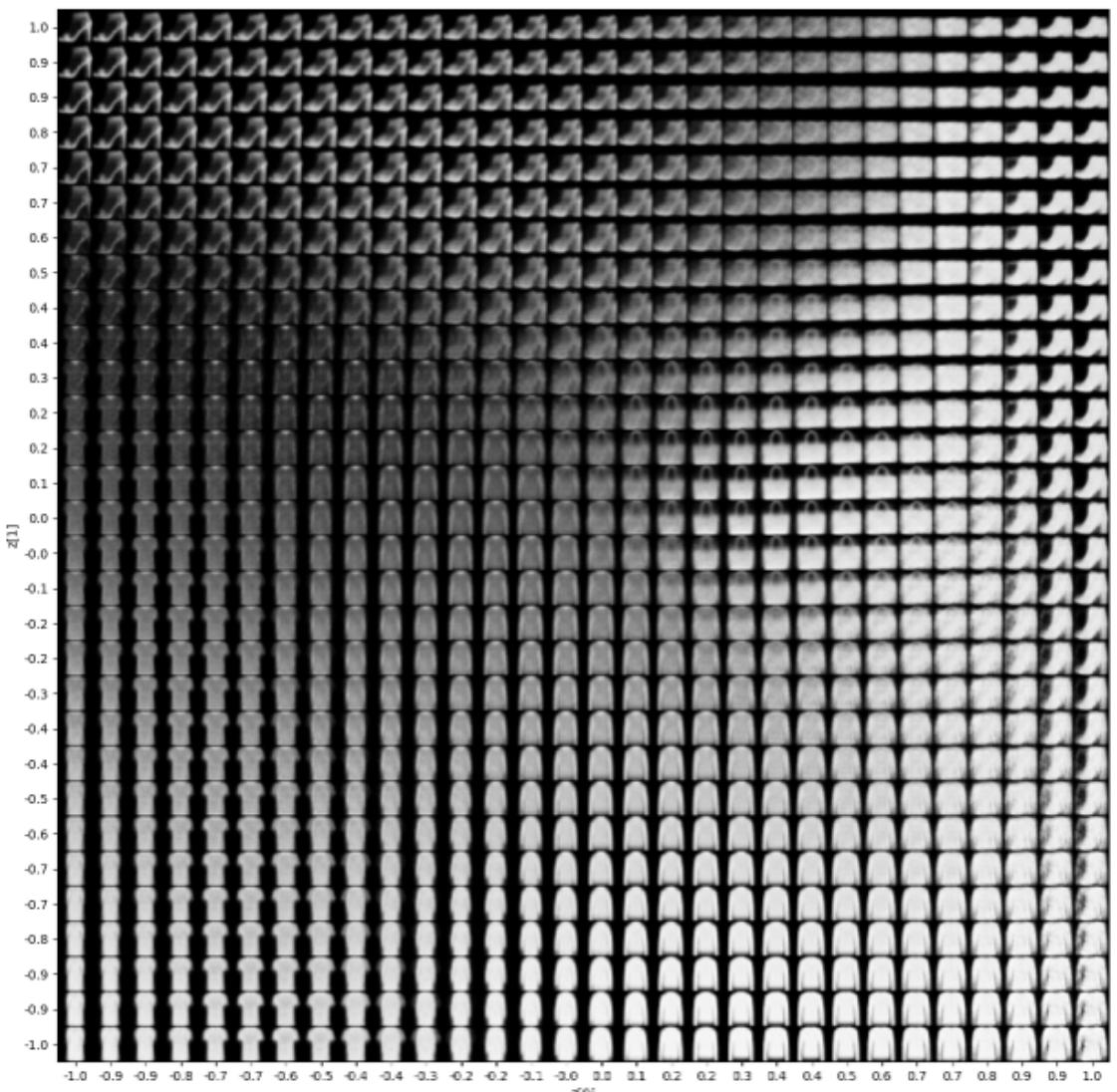
نتیجه برای حالتی که فضای latent اندازه 10 دارد به صورت زیر است:

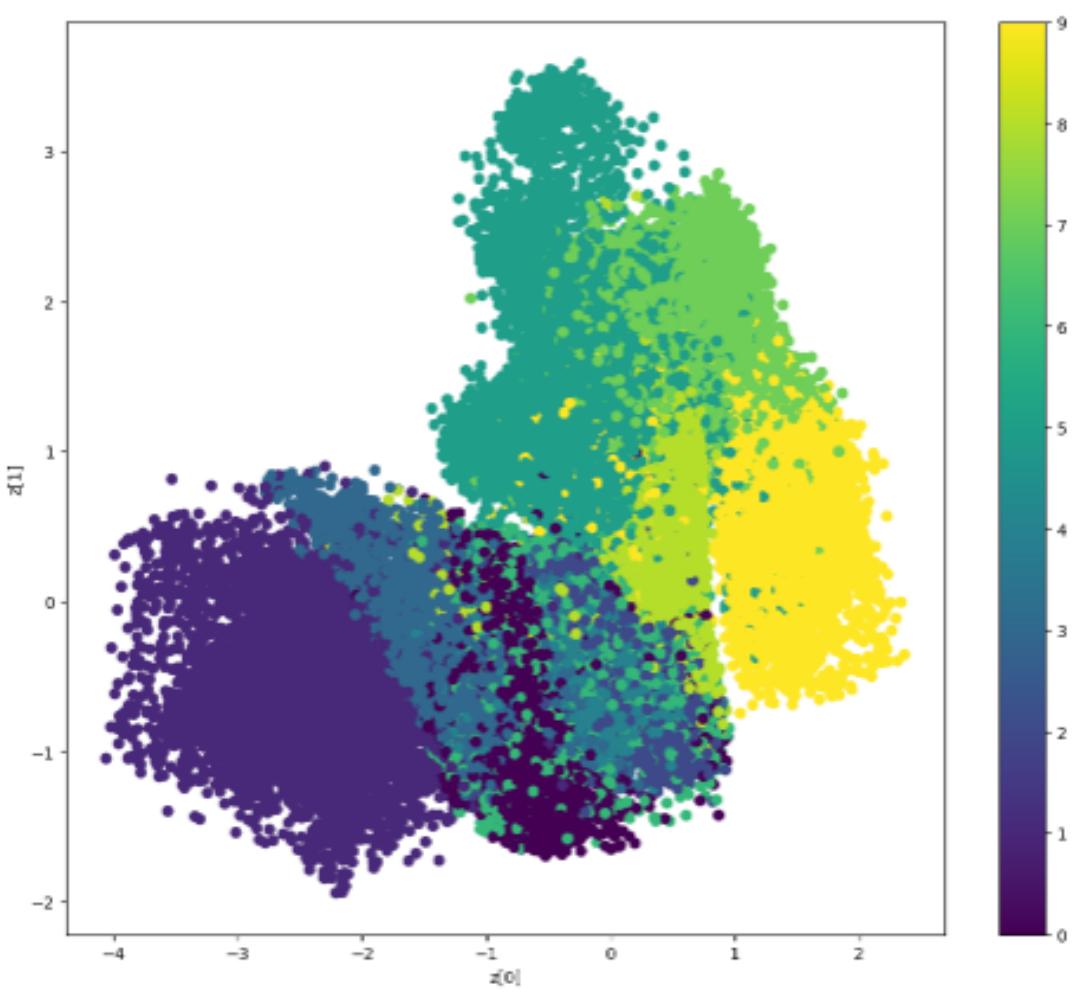


همچنین scatter plot مربوطه نیز به صورت زیر خواهد بود:



برای حالتی که بعد فضای latent برابر 2 هست نیز داریم:





Diffusion Models -6-۱

شبکه‌های مولد مبتنی بر Diffusion Model یک رویکرد نوین در حوزه تولید داده‌ها هستند که به تازگی محبوبیت بالایی پیدا کرده‌اند. این شبکه‌ها با استفاده از مدل‌سازی فرآیند انتشار گستته (discrete diffusion process) بر روی داده‌ها، توانایی تولید داده‌های جدید را دارند. در ادامه، شیوه عملکرد Diffusion Model و مقایسه آن با VAE را بررسی خواهیم کرد:

شیوه عملکرد Diffusion Model :

1. آغاز از نمونه‌گیری نمونه‌های تصادفی از توزیع احتمالی ساده.
2. اجرای چندین مرحله انتشار گستته در طول زمان. در هر مرحله، نمونه‌ها بر اساس تابع گستته‌ای به نام نمونه‌برداری از داخلی مدل به روز می‌شوند.

3. پس از مراحل انتشار، نمونه‌ها به داده‌های جدید تبدیل می‌شوند که می‌توانند به عنوان خروجی شبکه مولد استفاده شوند.

مقایسه با :VAE

- مزایا Diffusion Model :

- کیفیت تصاویر: Diffusion Model ها معمولاً توانایی تولید تصاویر با کیفیت بالا را دارند و نمونه‌های تولید شده توسط آنها واقع‌گرایانه‌تر و واضح‌تر هستند.

- عدم نیاز به فضای نهان: در Diffusion Model ها، نیاز به استفاده از فضای نهان برای تولید داده‌ها وجود ندارد.

- معایب Diffusion Model :

- پیچیدگی آموزش: آموزش مدل Diffusion Model ها پیچیده‌تر از آموزش VAE ها است و نیاز به محاسبات پیچیده‌تری دارد.

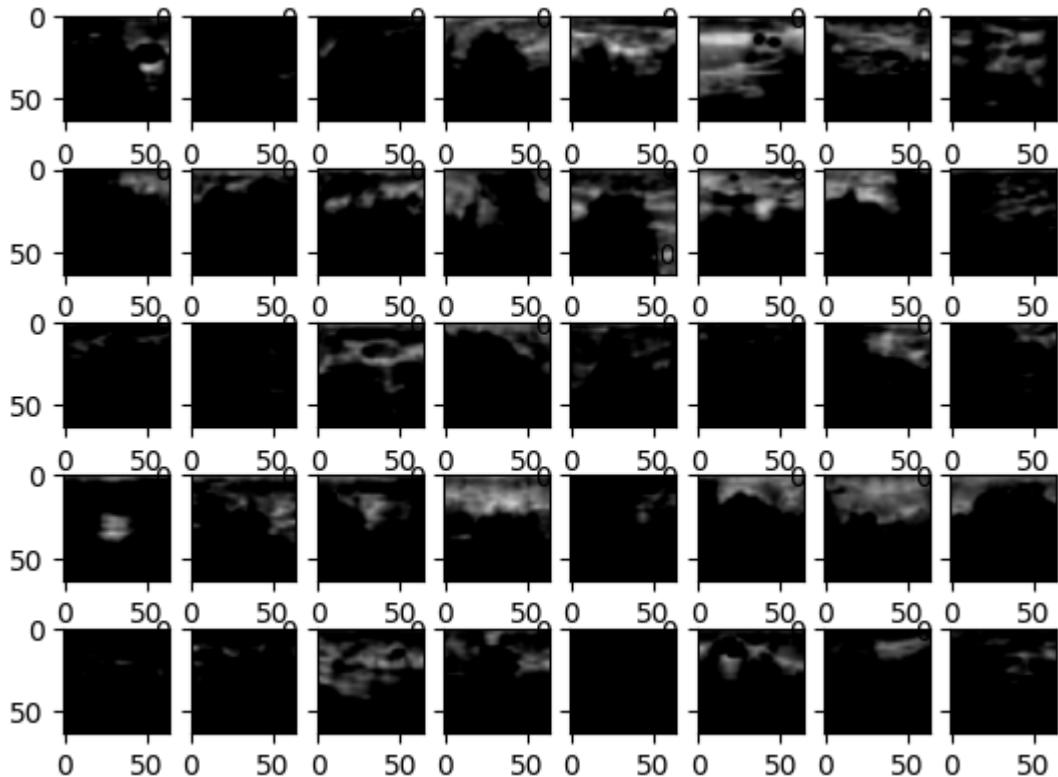
- منابع محاسباتی: Diffusion Model ها منابع محاسباتی بیشتری نسبت به VAE ها نیاز دارند.

در مقایسه با VAE ها، Diffusion Model ها بیشتر برای تولید تصاویر با کیفیت بالا و کاربردهای واقع‌گرایانه مناسب هستند. اما همچنین با پیچیدگی آموزش و نیاز به منابع محاسباتی بیشتر همراه هستند. در عوض، VAE ها برای کاهش ابعاد، تولید داده‌های جدید و کاربردهای گسترده‌تر عموماً مناسب‌تر هستند. انتخاب بین این دو روش بستگی به نیازها و هدف مورد نظر دارد.

پاسخ ۲ - شبکه متخصص مولد

۲-۱. بارگذاری داده ها و شبکه ی ResNet

پس از بارگذاری داده ها، چند تا از آن ها را به صورت رندوم نشان میدهیم:



شکل ۱. ۴۰ تصویر رندوم از BreastMNIST

>Loading Data

```
[7] train_dataset = BreastMNIST(split='train', transform=data_transform, download=download)
    val_dataset = BreastMNIST(split='val', transform=data_transform, download=download)
    test_dataset = BreastMNIST(split='test', transform=data_transform, download=download)
```

شکل ۱۲. بارگذاری داده ها

▼ Preprocessing

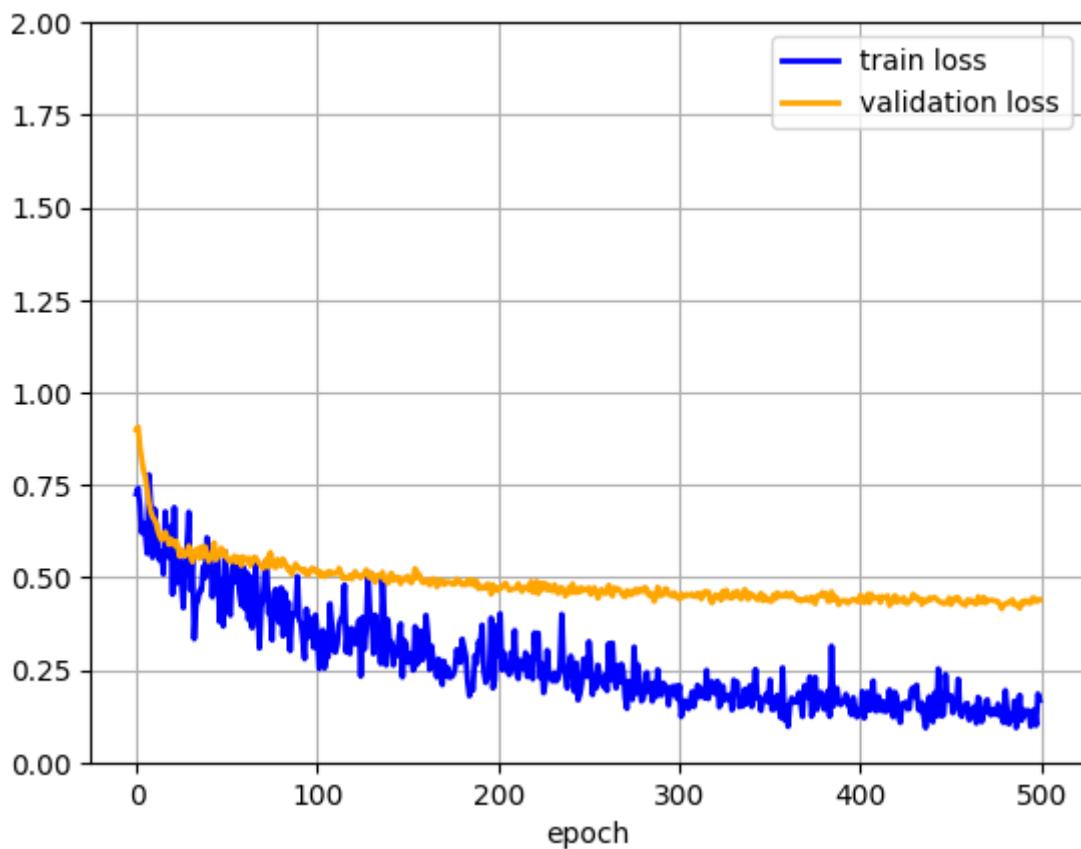
```
[6] IMAGE_SIZE = 64

data_transform = transforms.Compose([
    transforms.Resize(IMAGE_SIZE),
    transforms.ToTensor(),
    transforms.Normalize(mean=[.5], std=[.5]),
    transforms.Lambda(lambda x: x.repeat(3, 1, 1))
])
```

شکل ۱۳. پیش پردازش داده ها

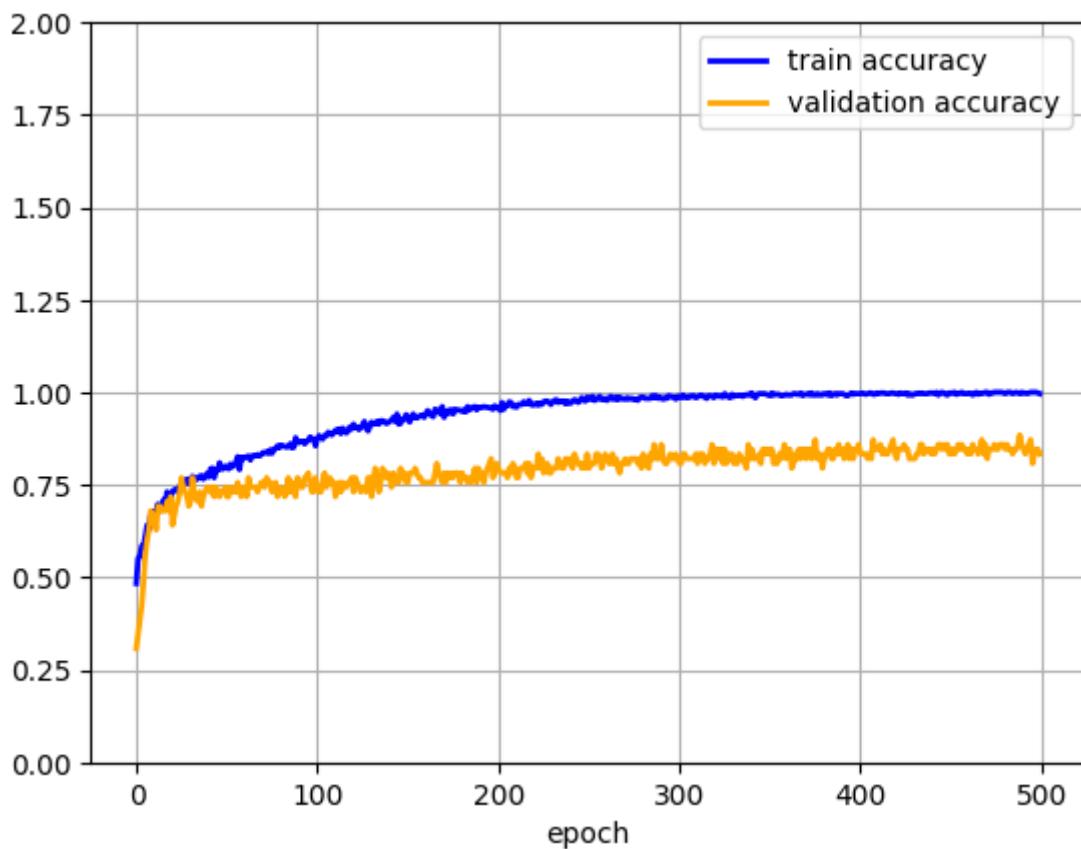
همانطور که در شکل ۱۳ مشاهده میشود تصاویر را رسایز کرده، به تنسور تبدیل میکنیم و سپس نرمالایز میکنیم با میانگین و واریانس ۰.۵.

(الف)



شکل ۲. نمودار خط روی داده های train و validation

مشاهده میشود با گذشت زمان از validation loss داده آموزشی و کاسته میشود و overfit رخداده است.



شکل ۳. نمودار دقت روى داده های train و validation

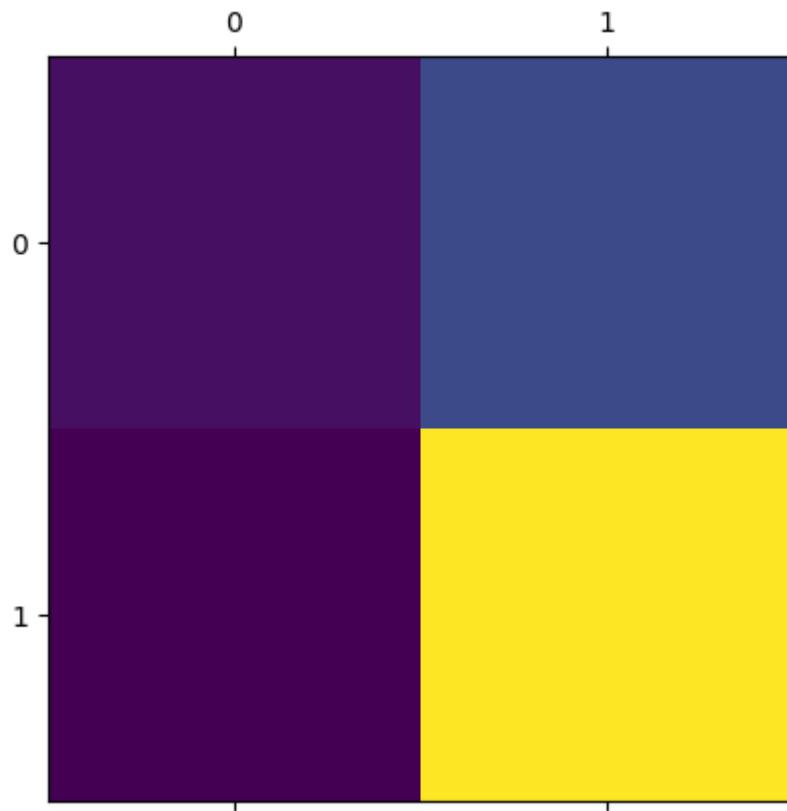
مشاهده ميشود دقت دادگان آموزشی به ۱ ميل کرد اما دقت validation پايين نيامد که نشان ميدهد overfit رخ نداده است و روى داده های validation نيز خوب عمل کرده است.

```
Eval Epoch 1: 100%|██████████| 1/1 [00:00<00:00,  8.91it/s, accuracy=0.756, loss=0.513]
```

شکل ۴. دقت روى دادگان test

مشاهده ميشود دقت روى دادگان test حدود 0.756 بوده است که با توجه به نمودار های شکل ۳ و ۴ قابل توجيه است.

(ب)



شکل ۵. ماتریش آشفتگی

ماتریس آشفتگی به صورت شکل ۵ میباشد که مقادیر دقیق هر بخش در شکل ۶ آمده است:

```
tensor([[ 12,  30],
       [  8, 106]])
```

شکل ۶. ماترس آشفتگی دقیق تر

بیشتر موارد درست متعلق به TN بوده اند و اکثر موارد اشتباه تشخیص داده شده FP بوده اند.

CGAN شبکه ۲-۲

(الف)

```
[ ] class cDCGAN_Generator(nn.Module):
    def __init__(self, z_dim, channels_img, features_G):
        super(cDCGAN_Generator, self).__init__()
        self.gen = nn.Sequential(
            self._block(
                z_dim,
                features_G*16, 4, 1, 0
            ),
            self._block(
                features_G*16,
                features_G*8, 4, 2, 1
            ),
            self._block(
                features_G*8,
                features_G*4, 4, 2, 1
            ),
            self._block(
                features_G*4,
                features_G*2, 4, 2, 1
            ),
            nn.ConvTranspose2d(
                features_G*2,
                channels_img,
                kernel_size=4,
                stride=2,
                padding=1
            ),
            nn.Tanh()
        )
    def _block(self, in_channels, out_channels, kernel_size, srtide, padding):
        return nn.Sequential(
            nn.ConvTranspose2d(
                in_channels,
                out_channels,
                kernel_size,
                srtide,
                padding,
                bias=False
            ),
            nn.ReLU()
        )
    def forward(self, x):
        return self.gen(x)
```

شکل ۷. ساختار کلی مولد

مطابق شکل ۷ مولد به صورت چهار بلوک کانولوشن ۲ بعدی با تابع فعال سازی ReLU ساخته شده است و در نهایت با تابع فعال ساز TanH فعال شده است. علت استفاده از لایه های کانولوشن به علت پردازش تصویر ها و خواص کلی کانولوشن هاست برای مثال ویژگی ها را هر کجا تصویر پیدا میکنند.

توجه شود از Convolution Transpose استفاده کرده ایم تا بتوانیم از ویژگی ها تصویر تولید کنیم. در بخش بعدی تصاویر را به فضای ویژگی میبریم تا بتوانیم آن ها را با یکدیگر مقایسه کنیم.

```
[ ] class cDCGAN_Discriminator(nn.Module):
    def __init__(self, channels_img, features_d):
        super(cDCGAN_Discriminator, self).__init__()
        self.disc = nn.Sequential(
            nn.Conv2d(
                channels_img,
                features_d,
                kernel_size=4,
                stride=2,
                padding=1
            ),
            nn.LeakyReLU(0.2),
            self._block(
                features_d,
                features_d*2, 4, 2, 1
            ),
            self._block(
                features_d*2,
                features_d*4, 4, 2, 1
            ),
            self._block(
                features_d*4,
                features_d*8, 4, 2, 1
            ),
            nn.Conv2d(features_d*8, 1, kernel_size=4, stride=2, padding=0),
            nn.Sigmoid()
        )
    def _block(self, in_channels, out_channels, kernel_size, stride, padding):
        return nn.Sequential(
            nn.Conv2d(
                in_channels,
                out_channels,
                kernel_size,
                stride,
                padding,
                bias=False
            ),
            nn.LeakyReLU(0.2)
        )
    def forward(self, x):
        return self.disc(x)
```

شكل ۸. ساختار کلی discriminator

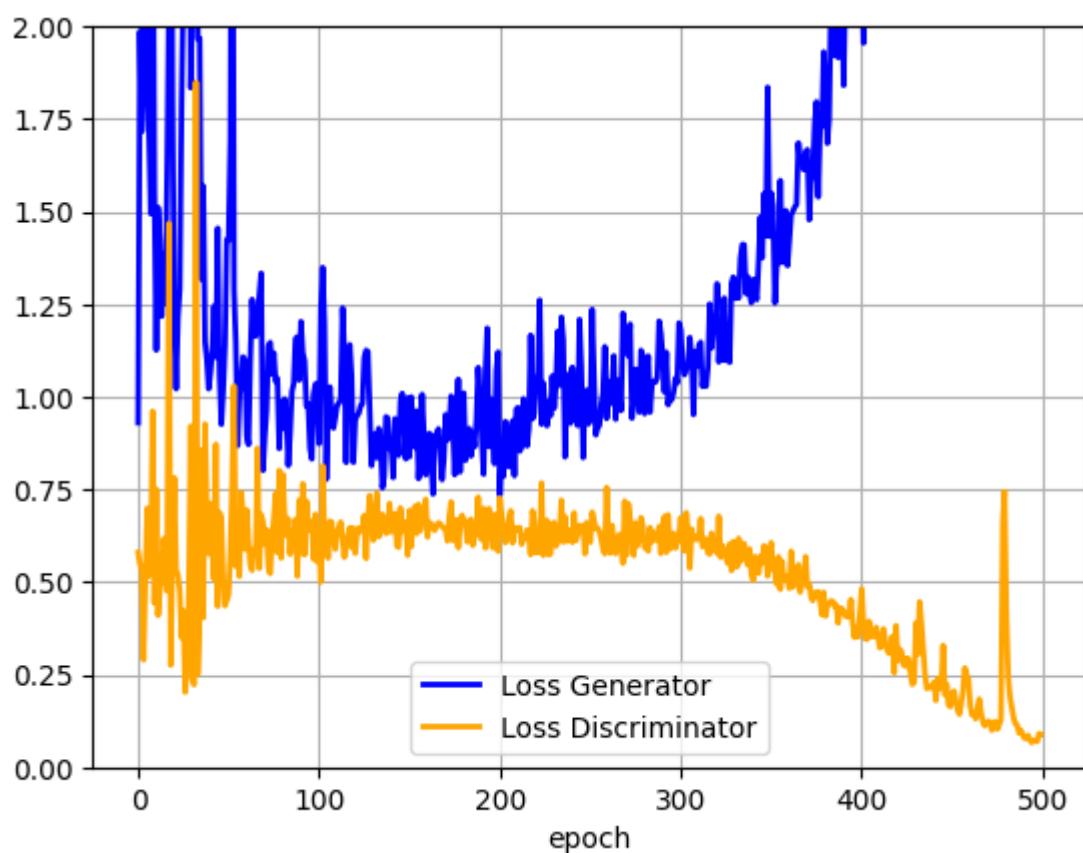
مشابه قسمت قبل از چهار بلوک کانولوشنی ساخته شده است که تابع فعال ساز هر بلوک مشابه است. در نهایت با تابع فعال ساز سیگموید فعال میشود.

از هایپر پارامتر های زیر برای آموزش شبکه استفاده شده است:

```
LearningRate = 2e-4
batch_size = 128
epochs = 500
features_discriminator = 64
features_generator = 64

image_size = 64
channels_image = 3
z_dimention = 100
```

ب) پس از آموزش ۵۰۰ ایپاک به نمودار لاس شکل ۹ میرسیم:

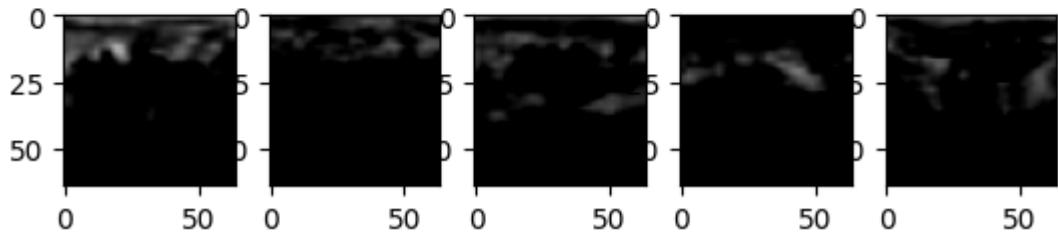
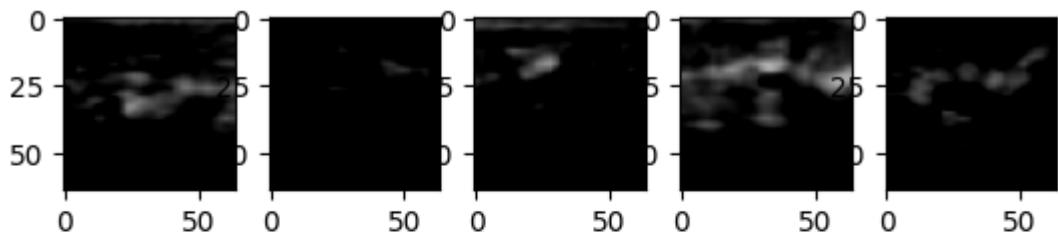


شکل ۹. نمودار خطای دو مژول Generator و Discriminator

طبق شکل ۹ تا ایپاک ۲۰۰ خطای مولد کم شده و discriminator زیاد شده است. این به این خاطر است که generator با تولید عکس های بهتر توانسته است به خوبی discriminator را فربیب دهد. اما از آنجایی که این شبکه متخاصل است به مرور زمان یادگیری شکل میگیرد و discriminator متوجه تصویر های تقلیبی از اصلی میشود. بعد از ایپاک ۲۰۰ خطای discriminator کاهش یافته و بهتر از مولد عمل میکند.

این تغییر حالت نشان میدهد یادگیری رخ داده است و شبکه به خوبی عمل کرده است. حالت ایده آل تر این است که هر دوی generator و discriminator در نهایت به یک نقطه converge کنند و با نوسان کمی جلو روند.

(ج)



شکل ۱۰. تصویر متعلق به کلاس ۱ جنریت شده توسط generator

(د)

انتخاب مناسب معماری شبکه مولد: در شبکه های مولد، انتخاب معماری مناسب بسیار مهم است. معماری هایی مانند GPT (مدل ترنسفورمر) معمولاً برای تولید متن و جملات بهتر عمل

می‌کنند. بنابراین، انتخاب معماری شبکه‌ای که به برازش بهتر با داده‌های ورودی و تولید خروجی با کیفیت کمک می‌کند، مهم است.

ترکیب با الگوریتم‌های پایدارسازی: استفاده از روش‌های پایدارسازی می‌تواند بهبود قابل توجهی در کیفیت خروجی شبکه‌های مولد داشته باشد. مثلاً می‌توانید از الگوریتم‌هایی مانند "Teacher Forcing" استفاده کنید که در آموزش شبکه مولد، از خروجی مطلوب به عنوان ورودی استفاده می‌کند و از احتمالات تولید شده توسط شبکه برای تولید بعدی استفاده نمی‌کند. این الگوریتم معمولاً بهبودی در تولید خروجی مولد دارد.

استفاده از تکنیک‌های تقویت یادگیری: تقویت یادگیری یک روش مطالعه رفتار مدل است که با استفاده از سیگنال تقویت، میزان کیفیت خروجی را بهبود می‌بخشد. این روش می‌تواند برای بهبود کیفیت خروجی مدل‌های مولد مفید باشد. با اعمال تکنیک‌های تقویت یادگیری می‌توانید شبکه مولد را آموزش دهید تا خروجی‌هایی با کیفیت واقعی‌تر و معنادارتر تولید کند.

استفاده از داده‌های بیشتر و تنوع بخشیدن به داده‌ها: آموزش شبکه مولد بر روی داده‌های بیشتر و تنوع بخشیده شده می‌تواند کیفیت خروجی را بهبود بخشد. با افزایش تنوع داده‌ها و آموزش شبکه بر روی مجموعه‌ای گسترده‌تر از داده‌ها، مدل بهبودهای قابل توجهی در تولید خروجی‌ها نشان خواهد داد.

استفاده از ارزیابی معیارهای کیفیت: برای ارزیابی و مقایسه خروجی‌های مدل‌های مولد مختلف، می‌توانید از معیارهای کیفیتی مانند CIDEr، ROUGE، BLEU و استفاده کنید. با استفاده از این معیارها، می‌توانید عملکرد مدل‌های مولد را مقایسه کنید و بهبودهای موردنیاز را تشخیص دهید.